Efficient DVFS to Prevent Hard Faults for Many-Core Architectures

Zhiquan Lai, Baokang Zhao, Jinshu Su

Efficient DVFS to Prevent Hard Faults for Many-core Architectures

Zhiquan Lai[1], Baokang Zhao[1], Jinshu Su[21]

[1]National University of Defense Technology, China
[2]National Key Laboratory of Parallel and Distributed Processing (PDL), China
{zqlai,bkzhao,sjs}@nudt.edu.cn

Abstract. *Dynamic Voltage and Frequency Scaling (DVFS)* is a widely-used and efficient technology for *Dynamic Power management (DPM)*. To avoid hard faults caused by voltage and frequency scaling, some overhead always be imposed on the performance of applications due to the latency of DVFS. Besides, on many-core architectures, the design of multiple voltage domains has made the latency of DVFS a much more significant issue. In this paper, we propose an efficient DVFS scheme to prevent hard faults, meanwhile eliminating the impact of latency of DVFS as possible. The main idea is applying *Retroactive Frequency Scaling (RFS)* where the latency of DVFS might be introduced. Based on the analysis, our approach is expected to achieve noticeable performance improvement on many-core architectures.

Keywords: efficient, DVFS, hard fault, many-core, retroactive frequency scaling (RFS).

1 Introduction

Low power has become a first-class design requirement in large-scale data centers and high performance computing systems. *Dynamic Voltage and Frequency Scaling (DVFS)* is a widely-used and efficient technology for *Dynamic Power management (DPM)* [1-3]. DVFS makes use of voltage scaling and frequency scaling to accomplish a trade-off between high performance and energy efficiency. Both of voltage scaling and frequency scaling take some latency to wait for voltage/frequency reach to given levels. However, the magnitudes of latency for voltage scaling and frequency scaling are much different. A frequency scaling only needs several CPU clock cycles, whereas a voltage scaling always needs tens, sometimes hundreds, of millisecond. Due to this difference, power state (voltage and frequency settings) could be in dangerous states (e.g. current voltage can't support the frequency) under some impropriate operations. In these cases, hard faults will happen and cause the CPUs to stop operating permanently [4].

Figure 1 shows the relationship between voltage and frequency during DVFS. As each CPU frequency setting has a least voltage value supporting the CPU operating in theory, we can draw a "Safe Boundary" in the frequency-voltage space. With this

boundary, all the voltage/frequency states could be classified into three categories. All the states above the boundary are dangerous because the voltages are not enough to support the frequency settings. The states just under the boundary are energy-efficient as the least voltages promise that no additional power is wasted. The states far below the boundary are non-power-efficient as some unnecessary voltage is wasted. For example, if we want to scale from *s0* to *s4* and scale up the frequency firstly, the frequency will reach *F1* from *F0* quickly. However, as the voltage can't scale up as soon as the frequency, the power state will be in dangerous state (like *s3* state). In this dangerous state, a hard fault will occur and the CPUs will stop operating permanently [4].

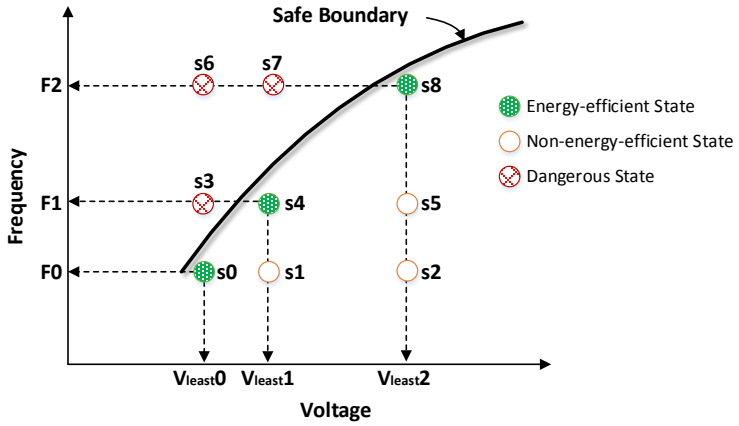

Fig. 1. Dangerous states causing hard faults during DVFS

To avoid hard faults caused by voltage and frequency scaling, some overhead always be imposed on the performance of applications due to the latency of DVFS. For example, as discussed in [5], in the cases of scaling up voltage and frequency, voltage is scaled firstly and the frequency will be not scaled until the voltage has reach the given level. And during voltage scaling, the application is stalled waiting the voltage scaling to finish. Although preventing hard faults, this type of DVFS schemes will cause some performance lost due to busy waiting within the latency of voltage scaling.

Besides, on many-core architectures, the design of multiple voltage domains has made the latency of DVFS a much more significant issue. As investigated in [5], the latency of voltage scaling could be up to 195ms in Intel SCC [6] many-core chip. The long latency of DVFS in many-core architectures could makes the DVFS scheme much more inefficient.

In this paper, we focus on the problem of inefficiency of DVFS due to latency of voltage scaling, especially for many-core architectures. We propose an efficient DVFS scheme to prevent hard faults, meanwhile eliminating the impact of latency of DVFS as possible. The main idea is applying *Retroactive Frequency Scaling (RFS)* where the latency of DVFS will be introduced. Instead of scaling frequency after

stalling the application during voltage scaling, we make the frequency scaling retroactive and keep the application running during voltage scaling.

The rest of the paper begins with an overview of related work in Section 2. Then we will introduce our efficient DVFS scheme with RFS methodology in Section 3. As we have not evaluated our approach using experiments yet, we analyze the benefit of our approach in Section 3.3. Section 4 summaries our work.

2 Related Work

DVFS on Many-core. DVFS technique is widely used for dynamic power management. However, in many-core architectures with tens or hundreds of cores in a single chip, e.g. Intel SCC, some new features of DVFS show up [5, 7, 8]. Rather than single voltage domain in one chip, multiple voltage domains are designed to separate cores into several independent power zones [6, 9]. Moreover, the latency of scaling voltage become much longer if scaling voltage simultaneously on multiple domains. Our previous work [5] investigated the feature of latency of DVFS on Intel SCC many-core platform, and proposed a latency-aware algorithm to avoid the aggressive power state transitions.

Reliability of DVFS. The negative effects of the DVFS technique on the system reliability have recently promoted the research on *reliability-aware power management (RAPM)*. A number of research works have already studied the effect of DVFS on reliability [4, 10-12]. Rosing et al. focused on the hard faults of *Systems on Chip (SoCs)* and studied the trade-off between reliability and power consumption [4]. Guo et al. study the RAPM problem for parallel real-time applications for shared memory multiprocessor systems in the presence of precedence constraints [11]. This paper will focus on the problem of inefficiency of DVFS due to latency of voltage scaling introduced for reliability. And the reliability in the paper focuses on the hard faults caused impropriate operation of DVFS.

3 Efficient DVFS with RFS

Considering the non-negligible latency of DVFS on many-core architectures, we propose an efficient DVFS using retroactive frequency scaling in this Section.

3.1 Case Study

Firstly, let us consider a case of scaling up voltage and frequency. As shown in Figure 2, it's the execution trace of a program running from T_0 to T_3[1]. At T_1, it decides to make a DVFS operation to scaling up the voltage and frequency. Assume the power state at from T_0 toT_1 is power efficient state (with frequency and its least voltage). As it needs to scale up frequency, it has to scale up the voltage firstly to avoid

[1] In this paper, T_i, i.e. T_1, T_2, T_3 and et al., denote specific timestamps in the execution trace of programs.

hard faults. However, the voltage scaling cost a non-negligible latency of T_2-T_1 so that it becomes safe to scale up the frequency. Thus, the execution of this program takes run time of T_3-T_0, including the latency of voltage scaling.

This is the traditional handling of voltage scaling and frequency scaling during DVFS. Obviously, this method is not efficient enough, especially when the latency of voltage scaling is non-negligible (e.g. the latency of voltage scaling on Intel SCC as investigated in [5]).

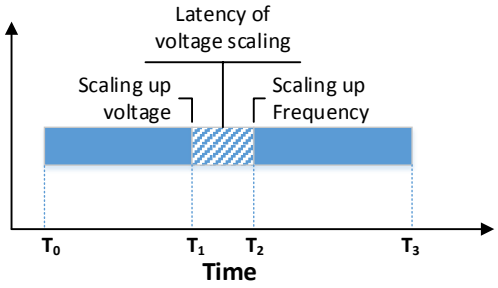

Fig. 2. Traditional voltage and frequency scaling up

3.2 Retroactive Frequency Scaling (RFS)

To eliminate the impact of latency of DVFS and improve the efficiency, we propose a novel DVFS scheme, named retroactive frequency scaling (RFS). As shown in Figure 3, let us review the case in last sub-section, RFS keeps the program running after scaling up voltage at T_1. Instead of stalling the program waiting for the voltage scaling up to given level, RFS makes the program running at the previous frequency setting during the latency of voltage scaling. When the voltage has reached the given level, we conduct the frequency scaling retroactively. By making use of the time slack during voltage scaling (from T_1 to T_2), RFS is capable to improve the performance of the program.

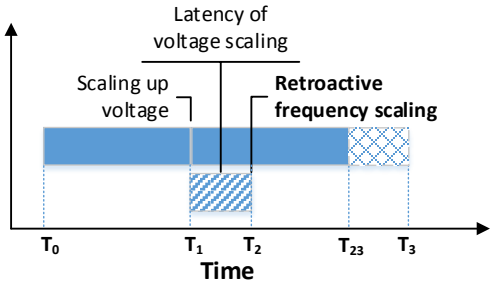

Fig. 3. Efficient voltage and frequency scaling up with retroactive frequency scaling

3.3 Analysis of Benefit

The benefit of RFS is obvious. With RFS, the program could keep running without any stall. As the latency of frequency scaling is in the scale of clock cycle, we don't consider it in this analysis. Thus, in the case shown in Figure 2 and Figure 3, the performance improvement (speedup) of RFS could be estimated by:

$$Speedup = 1 - \frac{(T1-T0)+(T2-T1)+(T3-T2)-(T2-T1)\cdot\frac{F_S}{F_d}}{(T1-T0)+(T2-T1)+(T3-T2)} \quad (1)$$

In Equation 1, the F_s denotes the frequency before scaling, and F_d denotes the frequency after scaling. Assuming the runtime performance (*1/execution-time*) is linearly with the frequency setting, $(T2-T1)\cdot\frac{F_S}{F_d}$ denoted the rough estimate of execution time saving due to applying RFS. The Equation 1 can be simplified into:

$$Speedup = \frac{T2-T1}{T3-T0}\cdot\frac{F_S}{F_d} \quad (2)$$

In Equation 2, T_2-T_1 means the latency of voltage scaling ($T_{latency\_volt}$), and T_3-T_0 means the whole execution time of the program under traditional DVFS ($T_{program}$). Hence, the equation can be denoted as:

$$Speedup = \frac{T_{latency\_volt}}{T_{program}}\cdot\frac{F_S}{F_d} \quad (3)$$

From the Equation 3, we can find that larger the latency of voltage scaling is, more speedup will be achieved from RFS methodology. In many-core architectures, as the latency of DVFS is non-negligible, our efficient DVFS scheme with RFS is expected to achieve noticeable performance speedup.

4 Summary and Future Work

In this paper, we propose an efficient DVFS scheme to prevent hard faults, meanwhile eliminating the impact of latency of DVFS as possible. Through our analysis, by applying Retroactive Frequency Scaling (RFS) methodology, we expect to achieve noticeable performance speedup. We are going to evaluate our approach on the Intel SCC many-core platform. According our previous work and experience done on this platform, we expect that our approach in the paper is feasible and efficient.

Acknowledgement. Special thanks to Intel China Center of Parallel Computing (ICCPC) in Wuxi City of China for providing the SCC hardware platform to support this research work. The work of this paper is also supported by the Program for Changjiang Scholars and Innovative Research Team in University (PCSIRT, No.IRT1012), and the Aid Program for Science and Technology Innovative Research Team in Higher Educational Institutions of Hunan Province.

References

1. D. Grunwald, I. Charles B. Morrey, P. Levis, M. Neufeld, and K. I. Farkas, "Policies for dynamic clock scheduling," presented at Proceedings of the 4th conference on Symposium on Operating System Design & Implementation - Volume 4, San Diego, California, 2000.
2. M. Weiser, B. Welch, A. Demers, and S. Shenker, "Scheduling for reduced CPU energy," in *Proceeding of the 1st USENIX conference on Operating Systems Design and Implementation*, Monterey, California, 1994.
3. D. Qingyuan, D. Meisner, A. Bhattacharjee, T. F. Wenisch, and R. Bianchini, "CoScale: Coordinating CPU and Memory System DVFS in Server Systems," in *Proceeding of the 45th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)* 2012, pp. 143-154.
4. T. S. Rosing, K. Mihic, and G. De Micheli, "Power and Reliability Management of SoCs," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 15, pp. 391-403, 2007.
5. Z. Lai, K. T. Lam, C.-L. Wang, J. Su, Y. Yan, and W. Zhu, "Latency-Aware Dynamic Voltage and Frequency Scaling on Many-core Architectures for Data-intensive Applications," presented at International Conference on Cloud Computing and Big Data (CloudCom-Asia), 2013.
6. J. Howard, S. Dighe, S. Vangal, G. Ruhl, N. Borkar, S. Jain, *et al.*, "A 48-Core IA-32 message-passing processor in 45nm CMOS using on-die message passing and DVFS for performance and power scaling," *IEEE Journal of Solid-State Circuits*, vol. 46, pp. 173-183, 2011.
7. S. Borkar, "Thousand core chips: a technology perspective," in *Proceedings of the 44th annual Design Automation Conference*, San Diego, California, 2007, pp. 746-749.
8. K. Ma, X. Li, M. Chen, and X. Wang, "Scalable Power Control for Many-Core Architectures Running Multi-threaded Applications," in *Proceeding of ACM/IEEE International Symposium on Computer Architecture (ISCA)*, San Jose, California, USA., 2011.
9. M. Gamell, I. Rodero, M. Parashar, and R. Muralidhar, "Exploring cross-layer power management for PGAS applications on the SCC platform," in *Proceedings of the 21st international symposium on High-Performance Parallel and Distributed Computing*, Delft, The Netherlands, 2012, pp. 235-246.
10. J. Haase, M. Damm, D. Hauser, and K. Waldschmidt, "Reliability-Aware Power Management of Multi-Core Processors," in *From Model-Driven Design to Resource Management for Distributed Embedded Systems*. vol. 225, B. Kleinjohann, L. Kleinjohann, R. Machado, C. Pereira, and P. S. Thiagarajan, Eds., ed: Springer US, 2006, pp. 205-214.
11. Y. Guo, D. Zhu, and H. Aydin, "Reliability-Aware Power Management for Parallel Real-time Applications with Precedence Constraints," in *the Second International Green Computing Conference (IGCC)*, Orlando, FL, 2011.
12. Y. Guo, D. Zhu, and H. Aydin, "Efficient Power Management Schemes for Dual-Processor Fault-Tolerant Systems," in *The First Workshop on Highly-Reliable Power-Efficient Embedded Designs (HARSH)*, Shenzhen, China, 2013.